DW_square

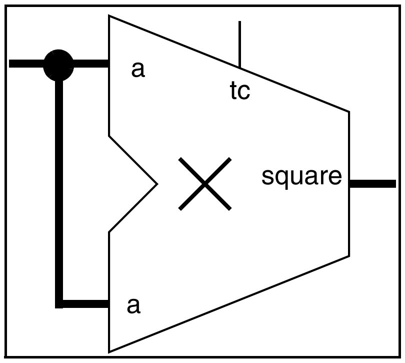
Integer Squarer

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Parameterized word length
- Unsigned and signed (two's complement) data operation
- Inferable using a function call



Description

DW_square is an integer numeric squarer.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*width* bits	Input	Input data
tc	1 bit	Input	Two's complement control ■ 0 = Unsigned ■ 1 = Signed
square	2 × *width* bits	Output	Product of (a × a)

Table 1-2 Parameter Description

Parameter	Values	Description
width	≥ 1 bits Default: 8	Word length of a

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
pparch[a]	Delay-optimized flexible Booth Wallace	DesignWare
apparch[a]	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

a. The ‘pparch’ (optimized for delay) and ‘apparch’ (optimized for area) implementations are dynamically generated to best meet your constraints. The ‘pparch’ and ‘apparch’ implementations can generate a variety of multiplier (squarer) architectures including Radix-2 non-Booth, Radix-4 non-Booth, Radix-4 Booth recoded and Radix-8 Booth recoded. The ‘pparch’ and ‘apparch’ implementations are generated making use of any special arithmetic technology cells that are found to be available in your target technology library. The `dc_shell` command, `set_dp_smartgen_options`, can be used to force specific multiplier (squarer) architectures. For more information on forcing generated arithmetic architectures, use 'man set_dp_smartgen_options' (in dc_shell) to get a listing of the command options.

Table 1-4 Simulation Models

Model	Function
DW02.DW_SQUARE_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_square_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_square.v	Verilog simulation model source code

Table 1-5 Functional Description

tc	a	square
0	a (unsigned)	Product of a × a (unsigned)
1	a (two’s complement)	Product of a × a (two’s complement, but always positive)

DW_square multiplies the operand `a` by itself to generate the product `square`. The control signal (`tc`) determines whether the input and output data is interpreted as unsigned (`tc` = 0) or signed (`tc` = 1) numbers.

Since the square of any number is always a positive value, the sign of the result of a signed square (the most significant bit of the output `square`) is always zero when DW_square is performing signed squaring (`tc` = 1).

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Function Inferencing - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_arith.all;

entity DW_square_func is
  generic ( func_width : NATURAL := 8 );
  port ( func_a      : in std_logic_vector(func_width-1 downto 0);
         func_tc     : in std_logic;
         square_func : out std_logic_vector((2*func_width)-1 downto 0) );
end DW_square_func;

architecture func of DW_square_func is
begin

  -- Functional inference of DW_square
  process (func_a, func_tc)
  begin
    if (func_tc = '0') then
      square_func <= std_logic_vector(DWF_square(unsigned(func_a)) );
    else
      square_func <= std_logic_vector(DWF_square(signed(func_a)) );
    end if;
  end process;
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW_square_func( func_a, func_tc, square_func );

  parameter func_width = 8;
  // Pass the width to the function
  parameter width = func_width;

  // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
  // to your .synopsys_dc.setup file (for synthesis) and add
  // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
  // (for simulation).
  `include "DW_square_function.inc"

  input [func_width-1 : 0] func_a;
  input func_tc;
  output [(2*func_width)-1 : 0] square_func;

  // Funtional inference of DW_square
  assign square_func = (func_tc == 1'b0)?
    DWF_square_uns(func_a) : // for unsigned input
    DWF_square_tc(func_a);   // for signed input
endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_foundation_comp.all;

entity DW_square_inst is
  generic ( inst_width : NATURAL := 8 );
  port ( inst_a      : in std_logic_vector(inst_width-1 downto 0);
         inst_tc     : in std_logic;
         square_inst : out std_logic_vector((2*inst_width)-1 downto 0) );
end DW_square_inst;

architecture inst of DW_square_inst is
begin

  -- Instance of DW_square
  U1 : DW_square
    generic map ( width => inst_width )
    port map ( a => inst_a,   tc => inst_tc,   square => square_inst );
end inst;

-- pragma translate_off
configuration DW_square_inst_cfg_inst of DW_square_inst is
  for inst
  end for; -- inst
end DW_square_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_square_inst( inst_a, inst_tc, square_inst );

  parameter width = 8;

  input [width-1 : 0] inst_a;
  input inst_tc;
  output [(2*width)-1 : 0] square_inst;

  // Instance of DW_square
  DW_square #(width)
    U1 ( .a(inst_a),   .tc(inst_tc),   .square(square_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
January 2019	DWBB_201806.5	■ Updated example in "HDL Usage Through Component Instantiation - VHDL" on page 5 ■ Added this Revision History table and the document links on this page